AMENDMENTS TO THE SPECIFICATION:

Please replace the paragraph appearing on pages 1-2 with the following amended paragraph:

[illegible] 9-29-03
[illegible] 9-27-03

A1

This application claims the benefit of the filing date of corresponding U.S. Provisional Patent Application No. 60/212,260, entitled MULTI-LAYER MAPPING TABLES, filed June ~~20~~ 19, 2000. ¶ In addition, the present invention is related to applications entitled A SYSTEM TO SUPPORT DYNAMICALLY FLEXIBLE DATA DEFINITIONS AND STORAGE REQUIREMENTS, serial no. 09/751,635, Attorney Docket Number 00-059-DSK, filed on December 29, 2000; USING CURRENT RECOVERY MECHANISMS TO IMPLEMENT DYNAMIC MAPPING OPERATIONS, serial no. 09/800,714, Attorney Docket Number 00-061-DSK, filed on March 8, 2001, <u>now U.S. Patent No. 6,532,527</u>; RECOVERY OF DYNAMIC MAPS AND DATA MANAGED THEREBY, serial no. 09/752,253, Attorney Docket Number 00-063-DSK, filed on December 30, 2000; FLOATING VIRTUALIZATION LAYERS, serial no. 09/752,071, Attorney Docket Number 00-116-DSK, filed on December 29, 2000; SELF DEFINING DATA UNITS, serial no. 09/751,641, Attorney Docket Number 00-117-DSK, filed on December 29, 2000; DYNAMICALLY CHANGEABLE VIRTUAL MAPPING SCHEME, serial no. 09/751,772, Attorney Docket Number 00-062-DSK, filed on December 29, 2000; APPARATUS AND METHOD FOR DYNAMICALLY CHANGEABLE VIRTUAL MAPPING SCHEME, serial no. <u>09/884,294</u>, Attorney Docket Number 00-060-DSK and APPARATUS AND METHOD FOR INSTANT COPY OF DATA IN A DYNAMICALLY CHANGEABLE VIRTUAL MAPPING ENVIRONMENT, serial no. <u>09/884,687</u>, Attorney Docket Number 2001-006-DSK, both of which are filed on even date hereof. All of the above related applications are assigned to the same assignee, and are incorporated herein by referenced.

A

Please replace the paragraph appearing on page 2, lines 10-16 with the following amended paragraph:

A2

The present invention relates generally to an improved data processing system and in particular to a data storage subsystem for use with a data processing system. Still more particularly, the present invention provides an apparatus and method for dynamically ~~changeable~~ changing the virtual mapping ~~scheme~~ of the data storage subsystem in a data processing system.

Please replace the paragraphs appearing on page 6, lines 3-27 with the following amended paragraphs:

A3

The present invention ~~provides a system~~ is directed to an apparatus and method to support dynamically changeable virtual mapping schemes in a data processing system. ~~The present invention separates p~~ Processing of data unit requirements in the present invention is separated from the selection of which storage subsystems to use for storage by using a storage methodologies inventory. A stored data management subsystem contains one or more hosts. A plurality of data storage elements is functionally coupled to the one or more hosts. The plurality of data storage elements is organized using a plurality of layers of mapping tables. The plurality of layers of mapping tables provides unique identification of location of the data such that individual data entries in a mapping table is variable and self-defining with respect to the amount of data managed.

In addition, the ~~present invention provides~~ various instant copy mechanisms are provided for copying data upon receiving a write operation to either original or to copy data. The instant copy mechanisms may be selected based on the type of mapping originally used to store the data that is to be copied. Other features and advantages of the present invention will be described in, or will become apparent to those of ordinary skill in the art in view of, the following detailed description of the preferred embodiments.

A

Please replace the paragraphs appearing on page 9, line 14 to page 10, line 9 with the following amended paragraphs:

Figure 18A, B, and C is an exemplary diagram illustrating an instant copy of old data when a write operation is performed on the original data;

9/28/03 Sub B1

~~Please replace the paragraph appearing on pages 9-10, lines 14+ with the following amended paragraphs:~~

Figure 20A and B is an exemplary diagram illustrating how the copy operations described with regard to **Figures 18A, B and C** and **19** may be used with variable size mapping units;

A4

Figure 23~~A~~ illustrates how the ~~present invention may be extended~~ original and copy data are moved to multiple ~~copy~~ physical storage areas;

~~**Figures 23B and C** show how the same operation as in **Figure 23A** may be performed for various amounts of new data in each of the two copy areas; and~~

Figures 24A, **B** and ~~**24B**~~ **C** illustrate an instant copy method~~, method C,~~ that may be used to copy data which was originally mapped using a full pointer system, such as log-structured file mapping~~.~~; and

Figures 25A and B illustrate an instant copy method that may be used to copy data which was originally mapped using a full pointer system.

Please replace the paragraph appearing on page 13, line to page 14, line 21 with the following amended paragraph:

A5

The present invention manages virtual storage facilities comprising an organization of computer equipment, for example, a host network, data transfer means such as fiber optic links, packet switched communication links, enterprise system connection (ESCON) fibers, small computer system interface (SCSI) cable, wireless communication links, storage controller means such as cache memory and shared memory, permanent storage means and attachment means connecting these devices together. The data storage facilities also may include management information associated with data units such that the management information provides an inventory of capabilities with upper and lower boundaries that may limit the options available to store the data and still meet a user's criteria. For purposes of this application, a data unit is a logical entity known to an owning entity that is composed of a number of data elements and meta-data, and a data element is a grouping of data bits or bytes that the subsystem chooses to manage as a consistent set. Such management information may be independent of attributes of or characteristics of the devices in the physical storage subsystem actually used to store the data elements, but may consist of imputed associations with those attributes through, for example, changeable rule sets, processes or algorithms. These rule sets, processes or algorithms may be changed by user demand or via processes, that may monitor data unit usage and manipulation. The storage of data elements may be adjusted to comply with modifications in, for example, the rules sets, processes or algorithms.

A5 cont'd

Please replace the paragraphs appearing on page 15, line 23 to page 16 line 14 with the following amended paragraphs:

Figure 3 is an exemplary block diagram of the conceptual relationship between the virtual device structures and the logical device structures in accordance with a preferred embodiment of the present invention. ~~The present invention provides a~~ A subsystem level application program interface (API) 312 is provided from host system 302 which allows a user to construct data unit definitions or virtual devices. These data unit definitions or virtual devices, such as, for example, virtual device structures 304, 306 and 308, may be called "Virtual Device Structures" (VDS). A subsystem in turn will implement logical device structures with mapping functions 310 and mapping VDSs into the physical world managed by the subsystem. The data may be mapped into networked storage subsystem 318 which may consist of logical device definitions 312, 314 and 316. Networked storage subsystem 318 may also consist of storage units 324 and 326 in which the data is stored. Also, data may be stored in technology storage subsystem 320 which may be a RAID and in physical storage devices 322.

A6

A

Please replace the paragraphs appearing on page 21, line 16 to page 22, line 22 with the following amended paragraphs:

A7

Figure 5 is a flowchart illustrating a data unit/virtual device structure data processing methodology in accordance with a preferred embodiment of the present invention. A top down approach may be used by building towards characteristics of known physical device types. For a collection of data elements with a virtual data unit address understood by host system ~~(step~~ **502~~)~~** the data unit virtual device structure address is processed (~~step~~ block **504**). The assigned virtual address communicated to the subsystem may be the same as or different from the virtual data unit address that is known to the host system. The data unit/VDS requirements interpretation is processed (~~step~~ block **506**), then the data units/VDS requirements are processed to map methodologies for implementation (~~step~~ block **508**). Then the storage subsystem selection processing for the data unit/VDS identifies which storage implementation methodologies are mapped to which potential subsystems, and selections for subsystem use are made (~~step~~ block **510**). Virtual data units are then communicated to the storage subsystem or subsystems (~~step~~ block **512**). Each storage subsystem creates a logical device structure to map the virtual data unit (~~step~~ block **514**).

Management interface **516** may manage data unit requirements inventory **522**, storage methodologies inventory **518** and receives and provides input from/to storage subsystem capabilities inventory **520**. Data unit requirements inventory receives input from data unit virtual device structure address processing (~~step~~ block **504**) and storage subsystem selection ~~in~~ processing data unit/VDS storage implementation methodologies when such methodologies are mapped to potential subsystems (~~step~~ block **510**). Storage methodologies inventory **518** receives input from data and provides input to data units/VDS requirements to ~~implement~~ map methodologies for implementation ~~processing~~ (~~step~~ block **508**).

Please replace the paragraph appearing on page 22, line 25 to page 23, line 6 and with the following amended paragraph:

A

With storage virtualization, a host server is freed from the restrictions of actual storage mechanisms. Furthermore, ~~the~~ an actual storage mechanism is freed from the restrictions of the presentation to the host server. Data storage is presented to the host server as an emulation of some device or media type or model. The data may actually be stored on one or more different types of devices and/or media. While storage management is concerned with physical characteristics of storage systems, devices and media, storage virtualization is concerned with masking the physical characteristics of storage systems and taking ~~the~~ control of these physical characteristics from the user or system administrator.

A8

Please replace the paragraph appearing on page 23, line 8 to page 24, line 10 with the following amended paragraph:

Figure 6 is an exemplary block diagram of the management API branch ~~illustrated in Figure 5~~ in accordance with a preferred embodiment of the present invention. In this example, Management API **610** may receive input from operations **602**, host **604** or through vendors updating existent capabilities **608**, which may be via a distributed data processing system, such as, for example, internet or web **606**. PARC system management **612** provides input and provides output to/from management API **610**. PARC system management **612** receives input from inventory of data unit requirements **614** along with inventory of storage methodologies **616** and inventory of storage subsystem capabilities **618**. Inventory of storage subsystem capabilities may be made up of existent storage subsystem capabilities **620** and installed storage subsystem capabilities **622**. If a data unit requirement or a storage methodology requires a particular storage subsystem capability, it needs to be determined as to whether the storage subsystem capability actually exists and, if so, whether the capability is actually installed on an available subsystem. If the storage subsystem capability is actually installed on an available subsystem, the required capability may be provided to satisfy data unit requirements **614** and/or implement a storage methodology **616**. However, if the data unit requirement or the storage methodology finds no capability existent within the inventory of storage subsystem capabilities, the data unit requirement and/or the storage

A9

A

A9 cont'd

methodology may request updates to subsystem capabilities **618** by way of vendors ~~update~~ updating existent capabilities **608**.

Please replace the paragraph appearing on page 28, lines 9-27 with the following amended paragraph:

A10

Figure 8 is an exemplary illustration of a hierarchical relationship of a mapping table in accordance with a preferred embodiment of the present invention. A map may be made up of several layers. These layers create a hierarchy used to facilitate the location of an entry that may describe the actual location of the data. Each layer points to a finer granularity than the layer below it. For example, the level 1 vector table **804** entries each describe, for example eight gigabytes of a Logical Unit Number (LUN) address space, each level 1 entry points to level 2 vector table **806** whose entries each describe, for example, eight megabytes of a LUN address space. The amount of space required to store layers **802** and **804** is small enough in this example so that dedicated memory may be set aside to hold them, thereby ensuring that any access requiring data stored in either level **802** or **804** will be found. Therefore, hits at these levels **(802 & 804)** will speed the processing of the mapping tables.

Please replace the paragraphs appearing on page 29, lines 1 to page 32, line 19 with the following amended paragraphs:

A11

In this example, LUN table **802** is a vector table with, for example, 256 entries, in which LUN table **802** is indexed by combining target and LUN addresses. There is one entry for each target and LUN combination. The entry contains a vector to the next table in level 1 vector table **804** layer or contains a null value if no target address ~~of~~ or LUN address has been created. LUN table **802**, in this example, requires 1024 bytes of memory. LUN table **802** may be pinned in memory.

Level 1 vector table **804** contains, in this example, 256 entries that represent the LUN eight gigabyte segments. Level 1 vector table **804** is indexed by using the most significant byte or bits 31-24 of the logical block address. Each entry either contains a

A

vector to level 2 table **806** or contains a null value. While the space to store all the level 1 table **804** pointers is reserved, in this example, for 256 entries, level 2 table **806** is only populated with enough entries to represent the size of the host LUN. That is, if the host LUN has a capacity of, for example, 50 gigabytes, there may be seven entries in level 1 vector table **804**. Level 1 vector table **804** requires, for example, up to 256~~K~~ kilobytes (kb) of memory. Level 1 vector table **804** is also pinned in memory.

A1

Level 2 vector table **806** contains, in this example, 1024 entries and is indexed by bits 23-14 of the logical block address. The entries in level 2 vector table **806** may contain either a pointer to a control block table or a null value. As each level 2 vector table **806** represents, for example, eight gigabytes of LUN address space, a null value may be present for address~~eds~~ that exceed the capacity of the LUN up to, for example, eight gigabytes of boundary. Each entry in level 2 vector table **806** represents, for example, eight megabytes of LUN address space. Level 2 vector table **806** may require, for example, 4096 bytes of memory and is pageable. Level 2 vector table **806** may have a higher priority than control block table **808** and may only be swapped out of the table memory space to make room for more table information when necessary (i.e., when no lower level table information is available to be swapped out).

The lowest layer, in this example, in the map is control block table **808**. Control block table **808** is made up of, for example, 256 control block entries. Bits 13-6 of the logical block address are used as an index into control block table **808**. Control block table, in this example, represents eight megabytes of the LUN address space. Each control block table **808** requires, for example, 4096 bytes of memory. Control block table **808** is pageable and may have a lower priority than level 2 vector table **806** and may be swapped out of the mapping table memory space to make room for more entries (e.g., other level **808** entries) before level 2 vector table **806** is swapped. Control block table **808** may be swapped on a LRU basis. **Figure 9** is an exemplary diagram of a portion of a mapping table describing an address range with four distinct sections in accordance with a preferred embodiment of the present invention. Hashing algorithms are a well known mechanism for storage subsystems to manage space and resolve an input address to a physical storage location. Hash algorithm 1 **902** and hash algorithm 2 **904** are serially implemented algorithms that may be used in a storage subsystem by a processor, such as

A

processors **210-224** in **Figure 2**. Hash algorithm 2 **904** may resolve to several sections. Each section may in turn be mapped using different mapping techniques, such as, for example, load point and offset section **906**, not allocated section **908** log-structured file (LSF) section **910** and RAID section **912**. These mapping techniques, load point and offset, log-structured files, using a pointer and length for unallocated space, RAID, and the like, are well known mapping techniques in the computer industry. The present invention is directed to a mechanism for combining mapping techniques and dynamically modifying the specific mapping technique used for a particular subset of data. Such dynamic modification may include changing the mapping technique from one technique to another, then to possibly ~~to~~ another, or perhaps back to a previously used technique.

A11

Figure 10 is an exemplary block diagram of a multi-layer mapping table in accordance with a preferred embodiment of the present invention. Hash algorithm 1 **1002** produces hash values as output. Logical address ~~in~~ **1006** is input into hash table **1002**. Hash algorithm 1 **1002** may not require modification due to dynamic changes to the mapping. Hash algorithm 1 **1002** may only need modification to reflect logical device definition changes. Hash value out **1008** is input into hash algorithm 2 **1004**. Hash algorithm 2 **1004** may be modified as the mapping changes. Hash value output **1010** from hash algorithm 2 **1004** is input into pointer table **1011**. Pointer table **1011** consists of pointer entries ~~**1012-N**~~ **1012-1016 and 1020-N**. The pointer changes as the mapping changes. Hash algorithm 2 **1004** and pointer table **1011** are held consistent with respect to range. Also included are mapping table endpoints **1028**, **1030**, **1032**, **1034**, **1036** and **N'**. Mapping table endpoints **1028-N'** can be omitted, for example, in the case of unallocated space. Mapping table endpoints **1028-N'** may be a single entry or a complex set of entries and further logic, for example, such as hash algorithm 3 **1034** which produces hash output **1040** which is an input to hash table **1044**. The total number of mapping table entry points may vary dynamically during use as the table entries are modified.

Please replace the paragraphs appearing on page 34, lines 3 to page 35, line 10 with the following amended paragraphs:

A

A12

In this example, a set of disk drives, such as, disk drives **1110**, **1120**, **1130**, **1140**, **1150** and **1160** are shown. In this example, disk drive **1110** has a capacity of 500 allocation units. The other disk drives **1120-1160** have a capacity of 1000 allocation units of space. A RAID stripe, for example, RAID stripe 1 ~~**1111**,~~ is a 4+1 RAID 4 group. Four allocation units for data, for example, stripe locations **1111**, **1121**, **1131** and **1141**, each on a different disk drive, are combined with 1 allocation unit for parity data at location **1151** to be written as one stripe. The data units are not required to be symmetrical in location, such as, occupying the same physical addresses on the disk. Each unit is maintained as a discrete unit and not tied geometrically to the other units.

Following the allocation and RAID stripe 1 write, four disk drives, for example, disk drives **1120**, **1130**, **1140** and **1150**, have 999 data allocation units free and available for allocation, while disk drive **1110** has 499 allocation units free and disk drive **1160** still has 1000 allocation units free. Disk drives **1110**, **1120**, **1130**, **1140** and **1150** each have 1 data allocation unit not available (i.e., allocated and written).

Then the next data write operation is initiated for stripe 2, also a 4+1 RAID level four group. The space management mechanism acquires five more data allocation units, such as, for example stripe locations **1122**, **1132**, **1142** and **1152**, four for data and one for parity data at stripe location **1161**. As a result, disk drive **1160** with 1000 units available is selected to provide one allocation unit. Following this allocation and RAID stripe write, disk drive **1110** has 499 units free for allocation, the next four disk drives, for example, disk drives **1120**, **1130**, **1140** and **1150** have 998 units free for allocation and drive **1160** has 999 units free for allocation.

Please replace the paragraphs appearing on page 38, line 1 to page 39, line 10 with the following amended paragraphs:

A13

Bytes 0-3 may contain in addition to LUN **1202**, Flags **1204** and Number of Blocks **1208** as described above, Parity LUN (PLUN) **1222**. In addition, Bytes 13-16 may contain Parity LUN Logical Block Address **1224** instead of unused bytes **1214-1220**. Parity LUN **1222** is the LUN address of the drive containing either the parity information for the RAID stripe or another drive in the mirrored set containing host

A

information. Parity LUN Logical Block Address **1224** is the logical block on PLUN **1222** that the parity is written to.

Another important feature of FlexRAID is the self-describing nature of data. Each stripe in the stripe contains information that describes the contents of the stripe and the stripe's relation to other members of the stripe.

A13 cont'd

A difference between FlexRAid and a conventional RAID system is that meta-data is associated with each stripe. As each data stripe is written, the meta-data is prefaced to the stripe. The meta-data written to the parity disk may be actually an XOR'ed parity of other pieces of meta-data. The meta-data may be, for example, one 512 byte block long and may contain the information as shown in **Figure 13** which is an exemplary meta-data block in accordance with a preferred embodiment of the present invention.

Figure 14 is an exemplary illustration of a default variable within boundary information in accordance with a preferred embodiment of the present invention. There may be a default extent size and a map 1401 of which entries are exceptions ~~1401~~. In this example, assume a layer n with table n_a and entry x. Also, assume a location xx in which a first extent begins. An extent is EE bytes in size, however, in this example, each extent is 100 sectors. Except as noted in an exception map **1401,** the location of the extent may be calculated up to an exception point. A map of storage such as map **1402**, is implied by the algorithm and may be calculated so map **1402** may not even be stored until exceptions exist.

Please replace the paragraph appearing on page 40, lines 6-18 following amended paragraph:

A14

Entry **1408** is 25 bytes in size. Therefore, data for entry **1408** will be stored beginning at 1100 in storage allocation map **1404** and ending at 1125 in storage allocation map **1404**. Then entry **1410** will be stored in map **1412.** Since the entries after entry **1408** are empty in map **1401**, data for nth entry **1410** will be stored in storage allocation **1404** at a location x times 100 from 1125 which is the end of entry **1408** in storage allocation **1404**. Nth entry **1410** is 200 bytes in size. Therefore, since nth entry

A

A14 cont'd

1410 is the last entry in map **1412**, storage allocation **1404** ends at a value of 1125 plus x times 100 plus the 200 bytes represented by nth entry **1410**.

Please replace the paragraph appearing on page 46, lines 12-25 following amended paragraph:

A15

For example, files 1, 2 and 3, as well as unallocated space, are viewed by the host system as load point and offset files, i.e. the files are accessible using load point and offset address mapping. In addition, file 1 has no policy constraints, file 2 is mirrored, and file 3 has the policy constraint 6-9's Reliability High Performance (The notion of a specific number "N" of 9's of reliability (e.g. 6-9's) (Six nine's) means that the probability of not losing a piece of data is 0.999... where "N" is the number of 9's following the decimal point. Therefore, a 6-9's reliability means a 0.999999 probability of no data loss). File 4 is viewed by the host system as a cylinder-head-record type of addressing file with no policy constraints.

Please replace the paragraph appearing on page 48, line 2 appearing following amended paragraph:

A16

Thus, in order to perform an instant copy operation, it is important to ~~identifying~~ identify the sections of virtual memory containing the data to be copied and the type of mapping mechanisms used for mapping the data to the virtual volume. The present invention provides a mechanism for performing these functions as well as selecting an appropriate copying technique for performing the instant copy operation.

Please replace the paragraphs appearing on page 49, lines 3-20 with the following amended paragraphs:

A17

Returning to **Figure 16**, once the extents of the data which is to be copied are parsed, the copy mechanism(s) for each data mapping type in the portion of data to be copied are then set up (step **1630**). In a preferred embodiment, for example, data mapped using load point offset mapping may be copied using one of the copy control methods

A

A17 contd

~~A1, A2...An~~ A1, A2, ..., An, as will be described in greater detail hereafter. Data mapped using mirroring may use copy control method B, as will be described in greater detail hereafter, and data mapped using pointer mapped mapping mechanisms, such as log-structured files, may make use of copy control method C, as will be described hereafter.

The instant copy procedure is then performed using the identified copy control mechanisms or methods (step **1640**) and the operation terminates. This operation may be repeated for each portion of data to be copied using the instant copy procedure.

Please replace the paragraph appearing on page 51, lines 10-17 with the following amended paragraph:

A18

The current state of the art, shown in **Figure 18B**, is exemplified by the StorageTek SVA 9500™. As shown in **Figure 18B**, just after the instant copy command is initiated, there exist two instances of the data map, the original data map **1860** and the copy data map **1870**. Both maps point ~~at~~ to the exact same physical space and will do so until changes are made to the original data or to the copy data.

Please replace the paragraph appearing on page 52, lines 14-25 with the following amended paragraph:

A19

When the instant copy operation is initiated, there is no initial copying performed. This is because both the original data **1820** and the copy data **1830** are identical and thus, only one set of data need be stored. Two separate items of data are required only when a write occurs to overwrite an item of the original data **1820**, or the copy data **1830**. Thus, ~~the~~ actual copying is not performed until a write occurs to the either the original data **1820** or the copy data **1830**. One of the items of data will be stored in the initial physical storage area **1840**, and the other in the additional physical storage area **1850**.

Please replace the paragraphs appearing on page 53, line 10 to page 56, line 21 with the following amended paragraphs:

A

Figure 19 is an exemplary diagram illustrating an instant copy of data when a write operation is performed on the original data 1920. As shown in Figure 19, when new original data 1910 is to be written to a location in the initial physical storage area 1940, the original data in this location is copied to the additional storage area 1950 as old copy data 1915. A new pointer P1 is generated and stored in a pointer table of the copy data map 1870 to point to the old copy data 1915 in the additional physical storage area 1950. Once the old copy data (original data) is copied from the initial physical storage area 1940 to the additional storage area 1950, the new original data 1910 may be written to the physical location that was occupied by the old copy data. Thus, the original pointer in the pointer table of the original data map 1960 pointing to the location of the original data 1920 now may be used to reference the new original data 1910, and the new pointer P1 may be used to reference the old copy data 1915 copied to the additional physical storage area 1950.

A20

Figure 20A is an exemplary diagram illustrating an instant copy of data when a write operation is performed to the copy data 2030. As shown in Figure 20A, new copy data 2011 is to be written to the copy data 2030. In this case, the new copy data may be written to the copy data 2030 without having to copy data from the initial physical storage area 2040 to the additional physical storage area 2050 because any original data in the initial physical storage area 2040 and in the original data 2020 are the same. Since this data is the same, there is no need to copy it from one location to another. In this case a new pointer P2 is added to the pointer table of the copy data map 2070 that points to the new copy data written to the copy data 2030 in the additional physical storage area 2050.

As is shown in Figure 20B, a subsequent write of new copy data 2011A to the same copy data 2030 location (or to the original data 1910 in Figure 19) will result in no additional copy activity. The new data (copy or original) will be replaced by the new data and the pointer P1 updated to pointer P2 that points to the new new data.

As is shown in Figure 20B, the ~~The~~ above write operations may occur in combination with regard to the same original data 1920 and copy data 1930. That is, a first write operation may occur to the original data area 1920, thereby causing the old data in the original data 1920 to be copied to the ~~aditional~~ additional physical storage area

Received from < 9723672002 > at 9/8/03 5:33:12 PM [Eastern Daylight Time]

A

1950. The original pointer now references the new original data in the initial physical storage area 1940 and a new pointer P1 is generated to point to the old data in the additional physical storage area 1950. A second write operation 2011A may occur to the copy data 1930, which does not require that the old copy data 1915 in the additional physical storage area 1950 be copied. A new pointer P2 need only be generated in the meta-data to point to the new copy data 2011A in the additional physical storage area 1950.

A20

As mentioned above, each data map which constitutes the meta-data contains a pointer table having pointers for data in each of the original data and the copy data. The meta-data further includes information in correlation with the pointers in the pointer table indicating whether the pointers point to new or old data. The correlation information can be explicit as in setting new and old flags or can be implicit as in putting the pointers into a structure that identifies old from new.

Figure 21 is an exemplary diagram illustrating how the copy operations described above with regard to **Figures 19** and **20** may be used with variable size mapping units. As shown in **Figure 21**, the original data **2120** has had two sections of data overwritten by new original data **2110** and **2111**. As a consequence, the old data ~~**2115**~~ and ~~**2116**~~ that was in these two sections has been copied to the additional physical storage area **2150**. Furthermore, new pointers **P1** and **P2** were generated in the pointer table of the metadata to reference the old copy data **2115** and **2116** that was copied to the additional physical storage area **2150** as new copy data **2115** and **2116**.

In addition, one section of data in the copy data **2130** has been overwritten by new data ~~**2117**~~. As with the copy procedure described above in **Figure 19**, the original data in the section overwritten need not be copied to the additional physical storage area **2150**. Thus, only a new pointer **P3** ~~**2127**~~ is generated in the pointer table to point to the new copy data **2117** in the additional physical storage area **2150**.

The sections of old and new data may have variable size. For example, old (copy) data **2116** may be of a different size than old data **2117**. Thus, it is important to keep track in the meta-data of the size of each section of data pointed to by the pointers in the pointer table. Accordingly, the meta-data stores a data section size in association with the pointers in the pointer table. In this way, ~~and~~ any processor that makes use of the data

A

A20 cont'd

stored in the original physical storage area **2140** and/or additional physical storage area **2150** will be aware of the extent of the data sections.

Please replace the paragraph appearing on page 57, lines 5-19 with the following amended paragraph:

A21

Figure 22 is an exemplary diagram illustrating an instant copy method, i.e. method A2, in which all writes are made to the additional physical storage area **2250**. With this method, whenever a write operation is to be made to either the original data **2220** or the copy data **2230**, the write operation is only made to the additional physical storage area **2250**. Thus, as shown in **Figure 22** new original data **2211-2212** to be written to the original data **2220** is actually written to the additional physical storage area **2250**. Similarly, new new copy data **2217** that is to be written to the copy data **2230** is also written to the additional physical storage area **2250**. Thus, the additional physical storage area is a combination of new data for the original data **2220** and new data for the copy data **2230**.

Please replace the paragraphs appearing on page 59, line 7 to page 60, line 11 with the following amended paragraphs:

A22

This instant copy method can be extended to multiple copy areas, as shown in **Figure 24A**. The instant copy method, method A3, shown in **Figure 24A** illustrates how three virtual volume areas may be used to perform an instant copy. In this example, all new data to be written to the original data is written to a first additional physical storage area **2451** rather than the initial physical storage area **2440**. All new data to be written to the copy data is written to the second additional physical storage area **2450**. In the particular example shown in **Figure 24A**, as a result of the above write methodology, the original data **2420** contains 80% new data and the copy data **2430** contains 10% new data, the first additional physical storage area **2451** contains ~~the~~ 80% new original data, and the second additional physical storage area **2450** contains ~~the~~ 10% new copy ~~datadata~~ data.

A

When it becomes necessary to separate the various copies of data, the same movement methodology described above is used with the three data areas. That is, since the first additional physical storage area **2451** contains 80% new data for the original data, the old data is moved to the first additional physical storage area **2451**. Since the second additional physical storage area **2450** contains 10% new copy data, this new data **2417** is copied to the initial physical storage area **2440**. As a result, the initial physical storage area **2440** contains copy data and the first additional physical storage area **2451** contains the original data. **Figures 24B** and **24C** show how this same operation is performed for various amounts of new data in each of the two copy areas. This operation may further be expanded to embodiments using a copy method, method An, having more than two copy areas if so desired.

A22 cont'd

Please replace the paragraphs appearing on page 63, line 3 to page 64, line 7 with the following amended paragraphs:

Thus, for example, the range ~~2530~~ is identified in the instant copy meta-data as pointers **P1** and **P2** with an indication that these pointers refer to a range of pointers inclusively. Similarly, the range **2540** may be designated by the begin and end pointers **P3** and **P4**. Thus, the size of the pointer table in the instant copy meta-data is reduced.

A23

As shown in **Figures 18-25B**, the present invention provides various instant copy methods that may be used to perform an instant copy operation on data based on the mapping method originally used to store the data. The various instant copy methods provide varying levels of actual copying performed and amounts of meta-data that ~~needs~~ need to be maintained. Each method has its own advantages and disadvantages and thus, the use of each may be selected based on the particular application to which the present invention is placed. With all these instant copy methods, however, an efficient instant copy may be performed in view of the original mapping of the data to be copied.

It is important to note that while the present invention has been described in the context of a fully functioning data processing system, those of ordinary skill in the art will appreciate that the processes of the present invention are capable of being distributed in the form of a computer readable medium of instructions and a variety of forms and that

A

A23 amend

the present invention applies equally regardless of the particular type of signal bearing media actually used to carry out the distribution. Examples of computer readable media include recordable-type media such as a floppy disc, a hard disk drive, a random access memory (RAM) ~~RAM~~, compact disc read only memories (CD-ROMs) ~~CD-ROMs~~, and transmission-type media such as digital and analog communications links.

A